

Attorney Docket No.: 002489.P031

PATENT

UNITED STATES PATENT APPLICATION

For

COMPARAND GENERATION IN A CONTENT ADDRESSABLE MEMORY

INVENTORS:

Sandeep Khanna  
Jose Pio Pereira  
Sunder Raj Rathnavelu  
Ronald S. Jankov

Prepared By:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP  
12400 WILSHIRE BOULEVARD  
SEVENTH FLOOR  
LOS ANGELES, CA 90025-1026

(408) 720-8300

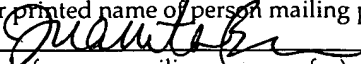
"Express Mail" mailing label number: EL351960963US

Date of Deposit: 10/31/01

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" under 37 C.F.R. § 1.10 on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D. C. 20231

JUANITA BRISCOE

(Typed or printed name of person mailing paper or fee)

  
(Signature of person mailing paper or fee)

10/31/01

(Date signed)

10000158-103101

# COMPARAND GENERATION IN A CONTENT ADDRESSABLE MEMORY

## FIELD OF THE INVENTION

[0001] This invention relates to the field of memory devices and, in particular, to content addressable memory devices.

## BACKGROUND OF THE INVENTION

[0002] Networks contain a collection of computing systems (e.g., clients and servers) that are interconnected by transmission lines to enable the transfer of data between them. A network typically includes multiple access points (e.g., routers and servers) that may switch and/or route data between transmission lines to transfer data from a source to a destination. Data is typically transmitted in the form of packets that are made up of smaller data cells. A packet is a unit of data that is routed between a source and a destination on a packet-switched network. When a file (e.g., e-mail, graphics, etc.) is sent from one place to another on a network, the file is divided into such smaller packets making them more efficient for transmission. The individual packets for a given file may travel different routes throughout networks with each packet containing both data and transmission information associated with the routing of data. As such, a packet may be described as having a payload containing the data, and one or more headers that contain the routing information (e.g., a destination address).

[0003] When all the packets have arrived at a destination, they are reassembled into the original file at the receiving end. Such a packet switching scheme is an efficient way to handle transmission on a connectionless network.

This is in contrast to a circuit switching scheme where a connection (e.g., a voice connection) requires the dedication of a particular path for the duration of the connection.

[0004] A router is a device (e.g., hardware, firmware, software) that determines the next network segment to which a packet should be forwarded towards its destination. A router may be positioned at points within a network or where one network meets another, referred to as a gateway. A router may create and maintain tables of the available routes and their conditions for use with other information to determine the best route for a given packet. Typically, a packet may travel through a number of network points having routers before arriving at its destination.

[0005] When a data packet arrives at the input of a router, several lookups may be performed to determine the subsequent handling of the packet, as illustrated in Figure 1. The lookups may include, for examples, where to send the packet next (Next Hop), the quality of service requirement (QoS), the Ethernet port address, etc. Consider, for example, a packet arriving at Router-A. Router-A needs to determine whether the packet is destined for local servers connected directly to Router-A, or if the packet should go to the next router on a route (Router-B) to a destination. Additionally, Router-A may assign a priority based on the destination address (DA) and the source address (SA) of the packet.

[0006] The packet header may first be parsed or processed to get the values from different fields (e.g., SA, DA, protocol type, QoS, etc) in order to

perform the various lookups. A packet classification lookup, for example, may be performed using SA, DA and other relevant fields in the packet header. The Next Hop lookup, for example, may also be performed to determine whether the packet is meant for local servers or for Router-B. If the packet is destined for Router-B, the packet is then put in a queue for Router-B. If the packet is destined for a local server (e.g., Server-1 or Server-2), then a media access control (MAC) lookup is performed to send the packet to the appropriate server. In the preceding example, three lookups are necessary for sending the packet on its way: Packet Classification, Next Hop, and MAC. However, often there are other lookups performed on the packet header, with the number of lookups exceeding five or more.

[0007] Routers may use processors and content addressable memory (CAM) devices to perform the various lookups on packets. As opposed to a random access memory (RAM) device, in which information is accessed by specifying a particular memory location address, the data stored in a CAM is accessed by the contents of the data. More specifically, instead of using an address to access a particular memory location, a CAM uses a key that contains a portion of the desired contents of a particular memory cell in the memory device. The CAM can be instructed by a processor to compare the key, also referred to as comparand data (e.g., packet header data) with data stored in its associative memory array, as illustrated in Figure 2. The CAM simultaneously examines all of its entries and selects the stored data that matches the key.

[0008] When the entire CAM device, or blocks thereof, is searched simultaneously for a match of the stored data with the key comparand data, the CAM device indicates the existence of a match by asserting a match flag. Multiple matches may also be indicated by asserting a multiple match flag. The CAM device typically includes a priority encoder to translate the matched location into a match address or CAM index. The priority encoder may also sort out which matching memory location has the top priority if there is more than one matching entry.

[0009] Data may be represented in the form of strings of binary digits ("bits") having a low ("0") logic state and a high ("1") logic state. Different types of CAMs may be used with different data formats. A binary CAM is designed to operate with "0" and "1" states, while a ternary CAM is designed to operate with "0", "1", and "don't care" states. The bits may be organized into groups such as a word (e.g., 64 or 72 bits wide) and stored in different segments of a CAM. The keys used for different data fields may have different word sizes, for example, the key for a Classification lookup may be 128 bits wide and the key for a Next Hop lookup may be 32 bits wide.

[0010] A router may include multiple CAMs, with each CAM having a different table or, alternatively, a single CAM having multiple blocks for each of the different tables, for performing the different lookups. For example, a router may include a 32 bit wide Next Hop CAM, a 128 bit Classification CAM, and a 48 bit MAC CAM. With routers having multiple CAMs, each of the multiple CAMs

are typically connected to common buses that are used to communicate the various keys and other input and output data with each of the CAM devices. Similarly, with routers having a single CAM with multiple blocks, each of the blocks is accessed using common buses. Thus, lookups are typically performed sequentially before a packet is processed (e.g., routed to the next destination or classified). Because the buses are shared with so many input and output functions of all the CAMs or CAM blocks, many clock cycles are required to multiplex data on the bus. This generally limits the search rate and overall throughput of conventional CAM devices. As the number of ports, segments, or devices that are supported by routers and as the number of lookups increase, conventional CAM devices and architectures can undesirably limit the system's overall throughput.

10000158-103101

## SUMMARY OF THE INVENTION

[0011] The present invention pertains to an apparatus and method of generating a comparand in a content addressable memory array. For one embodiment, the apparatus includes a content addressable memory (CAM) array and translation circuitry. The CAM array receives a comparand and the translation circuitry includes at least one first input, at least one second input, and at least one output. The first input is configured to receive an input data having a plurality of bit groups, wherein a first bit group has a first position in the input data relative to other bit groups. The second input is configured to receive translation information indicative of translation of the first bit group from the first position to a different position in a comparand. The output is coupled to the CAM array to transmit the comparand to the CAM array.

[0012] For one example, the translation circuitry includes a switch circuit that may include one or more multiplexers or demultiplexers. The translation circuitry may also include one or more storage elements to store the translation information, and one or more decode circuitry to decode the translation information and establish switch circuit connections between the first position and the position in the comparand. Other features and advantages of the present invention will be apparent from the accompanying drawings, and from the detailed description, which follows below.

## BRIEF DESCRIPTION OF THE DRAWINGS

[0013] The present invention is illustrated by way of example and not intended to be limited by the figures of the accompanying drawings.

[0014] Figure 1 is a conceptual illustration of packet handling by a router.

[0015] Figure 2 illustrates one embodiment of a CAM device.

[0016] Figure 3A illustrates one embodiment of a line card or blade of a router having a CAM device configured to decode of an input string according to the present invention.

[0017] Figure 3B illustrates one embodiment of input data in the form of an input string.

[0018] Figure 3C illustrates a CAM device having multiple blocks or arrays of CAM cells that are coupled to corresponding translation circuitry.

[0019] Figure 4A illustrates one embodiment of a CAM device having translation circuitry.

[0020] Figure 4B illustrates one embodiment of a switch that may be used in a CAM device.

[0021] Figure 5A illustrates an exemplary embodiment of a CAM device.

[0022] Figure 5B illustrates an alternative exemplary embodiment of a CAM device.

[0023] Figure 6A illustrates an alternative embodiment of a CAM device having multiple program registers.

10000158-103101



[0024] Figure 6B illustrates one embodiment of a CAM array having multiple blocks and multiple block segments.

[0025] Figure 7 illustrates one embodiment of a CAM device having a multiple block CAM array and multiple translation circuitry.

[0026] Figure 8 is an illustration of the multiple cycle operation of the CAM device of Figure 7.

10000158-103101

## DETAILED DESCRIPTION

[0027] In the following description, numerous specific details are set forth such as examples of specific, components, circuits, processes, etc. in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that these specific details need not be employed to practice the present invention. In other instances, well known components or methods have not been described in detail in order to avoid unnecessarily obscuring the present invention.

[0028] Embodiments of the present invention include various method steps, which will be described below. The steps may be performed by hardware components or may be embodied in machine-executable instructions, which may be used to cause hardware components (e.g., a processor, programming circuit) programmed with the instructions to perform the steps. Alternatively, the steps may be performed by a combination of hardware and software.

[0029] Embodiments of the present invention may be provided as a computer program product, or software, that may include a machine-readable medium having stored thereon instructions. The machine readable medium may be used to program a computer system (or other electronic devices) to generate articles (e.g., wafer masks) used to manufacture embodiments of the present invention. The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, magnet or optical cards, flash memory, or other

type of media / machine-readable medium suitable for storing electronic instructions.

[0030] The machine readable medium may store data representing an integrated circuit design layout that includes embodiments of the present invention. The design layout for the integrated circuit die may be generated using various means, for examples, schematics, text files, gate-level netlists, hardware description languages, layout files, etc. The design layout may be converted into mask layers for fabrication of wafers containing one or more integrated circuit dies. The integrated circuit dies may then be assembled into packaged components. Design layout, mask layer generation, and the fabrication and packaging of integrated circuit dies are known in the art; accordingly, a detailed discussion is not provided.

[0031] In one embodiment, the methods and apparatus described herein may be used to translate a input data (e.g., an input string) to multiple comparands for concurrent searching in a content addressable memory. In one embodiment, two or more comparands of differing widths may be generated concurrently from a common input string using multiple translation circuitry.

[0032] Each translation circuitry includes storage elements, for example, program registers that may be programmed with the translation information that determines the manner in which bytes from the input string are translated into the comparand. The translation information stored in the program registers may be decoded by decode circuitry. The decoded information is used to control the

operation of a switch to select one or more particular bits or bit groups of the input string for generation of a comparand designated by the decoded information. The resultant comparand string may be contiguous, or may have gaps, or may have repeated bits or groups of bits. The resulting comparand may be used to perform a look-up in an associated CAM array or smaller section thereof (e.g., a block or a block segment). In one embodiment, the comparand may be loaded into positions of a comparand register designated by the decoded information and stored before a look-up in an associated CAM array is performed.

[0033] It should be noted that while at times reference may be made to “bytes,” such reference is only exemplary for ease of discussion and, unless otherwise stated, is not meant to limit the invention. As such, the methods and apparatus discussed herein may be implemented with one or more bits or bit groups (with each bit group containing one or more bits). In addition, the steps and operations discussed herein (e.g., the loading of registers) may be performed either synchronously or asynchronously. The term “coupled” as used herein means connected directly to or connected through one or more intervening components or circuits.

[0034] In one embodiment where the width of the input data is larger than the width of the input bus on which the input data is applied to the CAM device, the translation circuitry may receive segments of the input data over multiple operation cycles. In such an embodiment, the translation circuitry may include

multiple program registers with each program register storing translation information for each segment of the input string received in the different cycles. By decoding common input data to generate multiple comparands in parallel, rather than sequentially, concurrent lookups on separate CAMs (e.g., arrays, blocks, block segments) may be performed and packet throughput in a CAM device may be significantly increased.

[0035] Figure 3A illustrates one embodiment of a line card or blade of a router having a CAM device configured to perform concurrent lookups. Line card 300 includes processor 310, ingress interface circuitry 330, egress interface circuitry 340, CAM device 320, associated data storage unit 370, traffic manager 360, and payload storage unit 350.

[0036] Processor 310 functions to control the overall operation of line card 300 in cooperation with the other components of line card 300. For example, processor 310 receives packets from a network medium through ingress interface circuitry 330, stores the payload of packets in payload storage unit 350, and processes packet header information to determine required lookups in CAM device 320 and subsequent handling of the packets, as discussed herein. Ingress circuitry includes, for example, PHY and MAC devices. Processor 310 sends out packets on a network medium through egress interface circuitry 340 based on the lookups performed by CAM device 320. Egress interface circuitry 340 may be connected to a switch fabric or directly to one or more other routers or switches. Processor 310 may be one or more network processor units (NPUs),

microprocessors, or one or more special purpose processors such as a digital signal processor (DSP). In another embodiment, processor 310 may be another type of controller, for example, a field programmable gate array or a general purpose processor. The processor 310, ingress interface circuitry 330, and egress interface circuitry 340 components of a router are known in the art; accordingly, a detailed discussion is not provided.

[0037] In response to information in a packet header, for a particular packet, processor 310 determines the number and types of lookups to be performed by one or more of CAM devices 320, and forms the search keys for these lookups. The searches or lookups may include, for example, Classification lookups, forwarding lookups (e.g., Next Hop or longest prefix match (LPM) lookup, MAC lookup, MPLS lookup, etc.). When multiple searches are required, processor 310 forms a composite search key that includes at least two, and as many as all, of the various search keys for the lookups. The composite search key is provided as a common input string to CAM device 320. CAM device 320 selectively identifies and extracts the individual search keys from the common input string and provides the individual search keys to an associated CAM array or block to perform a lookup. A block may be an entire array, sub-array, or a portion of an array or sub-array. Where CAM device 320 includes multiple arrays, blocks, or block segments, as discussed below, different lookups can then occur concurrently or simultaneously in CAM device 320, thereby increasing

100000158-103101

overall throughput over conventional systems in which searches are processed sequentially.

[0038] CAM device 320 includes translation circuitry to generate search keys from the common input string that are provided to the associated CAM array to perform the lookups, as discussed below. After one or more lookups are executed in CAM device 320, associated information for matching entries (e.g., additional routing information and/or packet information) may be retrieved from associated data unit 370. Processor 310 then communicates with traffic manager 360 to schedule the exit of a packet from line card 300 via egress interface circuitry 340.

[0039] Figure 3B illustrates one embodiment of input data in the form of an input string. In one embodiment, input string 405 may include field segments parsed or processed from one or more packet headers 311 and 312. When data processing systems (e.g., routers, clients, servers) exchange data over a network, the procedure involves the use of protocols by which these systems agree on how to communicate with each other. To reduce design complexity, networks may be organized as a series of layers. The number of layers and the function of each layer varies from network to network.

[0040] For example, where a transmission control protocol (TCP)/Internet protocol (IP) is used, it is organized into multiple layers including a network access layer and an Internet layer. The network access layer uses a TCP to enable the exchange of data between an end system and a network. An Internet layer

uses an IP to enable data to transverse multiple interconnected networks. Each of these protocols use packet headers containing routing information, as discussed above. For example, TCP packet header 311 includes a source address (SA) port segment 352 and a destination address (DA) port segment 353, and IP packet header 312 includes a SA segment 354, a DA segment 355, a type of service (ToS) segment 351, and a protocol type segment 356.

[0041] In one embodiment, for example, processor 310 of Figure 3A may be used to parse certain segments from packet headers 311 and 312 to generate input string 405 and transmit the input string to CAM device 320. For example, input string 405 may include MAC segment 357, TOS segment 351, SA port segment 351, DA port segment 352, SA segment 354, and DA segment 355. Alternatively, input string 405 may include more or less than the segments illustrated. One or more bits or group of bits (e.g., bytes) of different field segments of input string 405 may be translated to generate different comparand strings to concurrently perform different lookups in the CAM blocks, as discussed below. In an alternative embodiment, processor 310 may transmit as-received unparsed header segments to CAM device 320.

[0042] Figure 3C illustrates a CAM device having multiple blocks or arrays of CAM cells coupled to a corresponding translation circuitry. For example, CAM device 320 may include multiple CAM blocks 0, 1, and 2 with each block coupled to a corresponding translation circuitry 315, 316, and 317,



respectively. Each of the translation circuitry 315-317 is configured to receive and process input string 405.

[0043] In one embodiment, translation circuitry 315-317 may be pre-programmed to translate particular segments of the input string 405 in order to perform concurrent lookups on the various tables stored in blocks 0-2. For example: translation circuitry 317 may be preprogrammed to translate one or more bits or group of bits of MAC segment 357 to the comparand string 380; translation circuitry 316 may be pre-programmed to translate one or more bits or group of bits of DA segment 355 to the comparand string 381; and translation circuitry 315 may be pre-programmed to translate one or more bits or group of bits of SA segment 354, DA field segment 355 and TOS segment 351 to the comparand string 382. By translating one or more bits or group of bits of the different field segments from a common input string 405, in parallel, each of comparand strings 380-381 may then be used to perform the various lookups in the CAM blocks. For example: comparand string 380 may be used to perform a MAC lookup in CAM block 0; comparand string 381 may be used to perform a Next Hop (e.g., LPM) lookup in CAM block 1; and comparand string 382 may be used to perform a Classification lookup in CAM block 2.

[0044] In one embodiment, the lookups may all be performed concurrently. This may improve packet throughput in a router over routers utilizing prior CAM architectures. For example, if each lookup individually requires n clock cycles to perform, only a total of n clock cycles may be required

to perform all three lookups, rather than  $3n$  clock cycles, because the lookups are performed concurrently. Alternatively, one or more of the lookups may be performed sequentially. In yet another embodiment, some, but not all, of the lookups may be performed concurrently.

[0045] Figure 4A illustrates one embodiment of a CAM device having translation circuitry. In one embodiment, CAM device 400 includes a CAM array 410, a comparand register 460, and translation circuitry 415. CAM device 400 may be CAM device 320 of Figure 3A. CAM array 410 (and the other CAM arrays discussed herein) includes CAM cells that may be of any type of CAM cells including binary or ternary NAND and NOR based cells that may be formed from either volatile or non-volatile elements.

[0046] In one embodiment, translation circuitry 415 includes input bus 435, switch 430, decode circuitry 440, and program register (PR) 450. Program register 450 may be pre-programmed with translation information that determines how one or more bits or group of bits of input data (e.g., input string 405) are translated to a comparand. Alternatively, storage elements other than a register may be used to store the programmed translation information including volatile and non-volatile elements. The resultant comparand string may be used to perform a lookup in CAM array 410. Alternatively, in one embodiment, the translated one or more bits or group of bits of input string 405 may be stored in a comparand storage element, for example, comparand register 460. One or more

global masking circuits (not shown) may be coupled between comparand register 460 and CAM array 410 to enable global masking as discussed below.

[0047] Program register 450 is coupled to decode circuitry 440. Decode circuitry 440 decodes the information stored in program register 450. The decode circuitry 440 is coupled (e.g., connected directly to or through one or more intervening circuits) to switch 430. Decode circuitry 440 generates control signals for programming switch 430 based on the information stored in the program register 450. The switch 430 is coupled to receive input string 405 via the signal lines of bus 435. Switch 430 represents a matrix of intersections between the signal lines of input bus 435 and positions of comparand register 460. In one embodiment, switch 430 includes n number of multiplexers 470 each coupled to receive as inputs a group of m signal lines of bus 435, as illustrated in Figure 4B. In one embodiment, there may be n groups of m signal lines. Although, "m" may be used, hereafter, to indicate a byte, or 8 bits, any other number of bits may be used. Moreover, each group may have the same or a different number of signal lines.

[0048] It should be noted that although decode circuitry 440 is illustrated separately from switch 430, the operations of decode circuitry 440 may be incorporated into switch 430 or performed by a separate circuit. Similarly the operations of other illustrated components of the figures may be performed by separate circuits or incorporated within other circuits.

[0049] In one embodiment, switch 430 may be a cross-bar switch that operates on a per bit group (e.g., per byte) basis. For example, switch 430 may be configured to select particular bytes of input string 405 transmitted on signal lines 435 and load them into certain byte positions of comparand register 460. The bytes on signal lines 435 are selected using multiplexers 470 under the control of decode circuitry 440 and correspondingly output to a particular byte position of comparand register 460. In this manner, one or more bytes of input string 405 may be re-arranged to have a different byte position in a comparand string stored in comparand register 460 than its position in input string 405. In one embodiment, switch 430 may be programmed during normal operations of the CAM device 400, for example, by processor 310 in line card 300 of Figure 3A. Alternatively, switch 430 may be pre-programmed prior to normal operations of the CAM device 400. Registers and multiplexers are known in the art; accordingly a detailed discussion is not provided.

[0050] In one embodiment, decode circuitry 440 may also be configured to output one or more write enable (WE) signals 480 to comparand register 460. The write enable signals operate to control when comparand register 460 (or a segment thereof) loads the output of switch 430. For one embodiment, a separate write enable signal may be generated for each segment of comparand register 460 associated with the decoded output of switch 430. Alternatively, the write enable signals may be generated through other means, for example, by processor 310 of Figure 3A. In yet another embodiment, the write enable signals may be

generated in response to CAM device 400 receiving a write instruction or control signal, for example, from an instruction decoder or other control circuitry that receives and processes instructions or control information from another device such as processor 310 of Figure 3A.

[0051] Figure 5A illustrates an exemplary embodiment of CAM device 400 showing non-exhaustive exemplary embodiments of switch 430, decode circuitry 440, and program register 450. In the illustrated embodiment of Figure 5A, switch 430 receives nine groups of signals from signal lines 531-539 of input bus 405. Each group is coupled to each of the nine multiplexers 571-579.

Alternatively, switch 430 may include more or less than nine signal lines and nine multiplexers.

[0052] In the illustrated exemplary embodiment, decode circuitry 440 includes a corresponding number of decoders 441-449 with the outputs of each decoder coupled to a corresponding one of multiplexers 571-579. For example, the outputs of decoder 441 are coupled to control or select inputs of multiplexer 571 and the outputs of decoder 449 are coupled to control or select inputs of multiplexer 579. Decode circuitry 440 is configured to receive translation information from PR 450 and decode the received translation information in order to control the operation of switch 430.

[0053] Each decoder (e.g., decoders 441-449) of decode circuitry 440 is coupled to receive translation information (e.g., one or more bits) from a corresponding register position of PR 450 (e.g., register positions 451-459). PR

450 stores the translation information that is decoded by decode circuitry 440 to program switch 430. Decoders are known in the art; accordingly a detailed discussion is not provided herein.

[0054] In one embodiment, for example, PR 450 is a nine position register storing information that correlates to a particular byte of input string 405. The information may be stored, for example, in binary format. With such a format, only four bits are needed in each position to reference a particular byte of the nine bytes of input string 405. For example, a 0100 bit pattern may be programmed in register position 451 to designate byte 4 of input string 405, and a 0110 bit pattern may be programmed in register position 452 to designate byte 6 of input string 405. Continuing the example, decoder 441 decodes the bit pattern stored in register position 451 and, as a result, outputs control signals to multiplexer 571 to select the byte 4 data ("E") on input signal lines 535 for output to register position 461 of comparand register 460. Decoder 442 decodes the bit pattern stored in register position 452 of PR 450 and, as a result, outputs control signals to multiplexer 572 to select byte 6 data ("G") on input signal lines 537 for output to register position 462 of comparand register 460. Each of decoders 443-449 may output control signals to multiplexers 513-579, respectively, to select a byte on a particular byte line 531-539 (for output to corresponding comparand register positions 464-469) based on data stored in PR positions 453-459, respectively.

[0055] As such, PR 450 may be programmed to determine the manner in which the bytes of input string 405 are loaded into comparand register 460 using switch 430. The comparand string is then used to perform a look-up in CAM array 410.

[0056] In one embodiment, each of the decoders of decode circuitry 440 may also be configured to output a write enable (WE) signal on a write enable line. The write enable lines may be coupled to comparand register 460, for example, decoder 441 may output a WE signal on line 581 coupled to byte position 461 of comparand register 460. The write enable signal operates to control when comparand register 460 (or segment thereof) loads the output of a corresponding multiplexer. Comparand register 460 may then output its contents to CAM array 410 in order to perform a look-up. Alternatively, the write enable signal may be generated through other means, for example, with processor 310 of Figure 3A. Alternatively, the write enable signals may be generated by a control circuit such as an instruction decoder, for example, in response to a write or write and compare instruction that causes comparand register 460 to be selectively loaded by the translation circuitry with data from input bus 435. Decode circuitry 440 may also be configured to receive one or more clock signal(s) from a clock generator (not shown) to control the operation of the decode circuitry. As previously mentioned, the decoders 441-449 may be part of multiplexers 571-579 with the information stored in PR 450 provided directly as select signals to multiplexers 571-579.

[0057] For another embodiment, a single write enable signal may be provided to more than one segment of comparand register 460.

[0058] It should be noted again that in alternative embodiments the methods and apparatus discussed herein may also be implemented on a bit basis, rather than a byte basis, where density requirements for the CAM device are not too stringent. For example, input bus 435 may have n signal lines with each signal line couple to receive a bit of input string 405. A corresponding number of multiplexers may be used to select from among the bits of the input string based on the decoding of bit data programmed in program register 450.

[0059] For another alternative embodiment, one or more of multiplexers 571-579 does not have inputs to receive all of the groups of signal lines 531-539. For one example, multiplexer 571 has inputs coupled to receive all of the groups of signal lines 531-539, multiplexer 572 has inputs coupled to receive groups of signal lines 532-539, and so on. For this example, multiplexer 579 may not be needed at all, and comparand register position 469 may be directly connected to group signal lines 539. Other schemes may also be used.

[0060] Figure 5B illustrates another embodiment of CAM device 400 showing alternative embodiments of switch 430, decode circuitry 440, and program register 450. In one embodiment, program register 450 may store the position of the comparand register 460 to which an input byte of input string 405 will be loaded into. In such an embodiment, switch 430 includes demultiplexers 591-599. Each group of signal lines 531-539 is coupled to only one of



demultiplexers 591-599, respectively. Each of demultiplexers 591-599 is coupled to all of the positions of comparand register 460.

[0061] The decoders of decode circuitry 440 are coupled to a corresponding one of demultiplexers 591-599. For example, the outputs of decoder 441 are coupled to control or select inputs 586 of demultiplexer 591 and the outputs of decoder 449 are coupled to control or select inputs 589 of demultiplexer 599. Decode circuitry 440 is configured to receive translation information from PR 450 and decode the received information in order to control the operation of switch 430. In the embodiment illustrated in Figure 5B, PR 450 stores information that correlates to a particular position of comparand register 460. For example, translation information may be programmed in register position 451 to designate position 463 of comparand register 460 and translation information may be programmed in register position 459 to designate register position 464 of comparand register 460. In this manner, decoder 441 decodes the translation information stored in register position 451 and controls demultiplexer 591 to output the bits (e.g., "A") of byte 0 of input string 405 to register position 463 of comparand register 460. Similarly, decoder 449 decodes the translation information stored in program register position 459 and controls demultiplexer 599 to output the bits (e.g., "I") of byte 8 of input string 405 to register position 464 of comparand register 460.

[0062] As discussed above with respect to Figure 5A, each of the decoders may also be configured to output a write enable (WE) signal to comparand

register 460 to control when comparand register 460 (or segment thereof) loads the output of a corresponding demultiplexer 591-599.

[0063] Additionally, the decoders 451-459 may be part of demultiplexers 591-599 with the translation information stored in PR 450 provided directly as select signals to demultiplexers 591-599.

[0064] For another alternative embodiment, one or more of demultiplexers 591-599 does not have outputs coupled to each of the comparand register positions. For one example, demultiplexer 591 has outputs coupled to all of the comparand register positions, demultiplexer 592 has outputs coupled to all of the comparand register positions except the most significant (e.g., the left-most) positions, and so on. For this example, demultiplexer 599 may not be needed at all, and comparand register position 469 (e.g., the right-most position) may be directly connected to group signal lines 539. Other schemes may also be used.

[0065] The translation circuitry described above may also include more than one program register with each program register storing different (or the same) translation information for the decode circuitry. Each program register may selectively provide its translation information to the decode circuitry so as to form different comparands for different lookups. This may be particularly useful, for example, where the input bus is narrower (i.e., has less signal lines) than the total number of input bits in an input string such that multiple clock cycles are used to supply the total input string over the input bus to the CAM device. A separate one of the program registers may be selectively enabled for

each clock cycle so as to provide its translation information for a corresponding segment of the input string provided on the input bus at any one time. An exemplary embodiment of the alternative CAM device is shown in Figure 6A.

[0066] Figure 6A illustrates CAM device 700 that includes CAM array 710, comparand register 760, and translation circuitry 715.

[0067] Translation circuitry 715 includes M number of program registers 790-791 that correspond to the M segments of input string 705. Each of the M segments of input string 705 have a size (i.e., a number of bits or signals) equal to or smaller than Y, where Y is the number of signal lines of input bus 735. The program registers 790-791 are coupled to selection circuitry 780. In one embodiment, selection circuitry 780 may be a M:1 multiplexer (MUX). Each of the program registers 790-791 may be programmed to cause switch 730 to select particular bits or group of bits (e.g., bytes) of the input string segments 705 that are received by switch 730 on a different cycle of device operation. For an example, if the input string 705 is 288 bits in size and the input bus 735 is 72 bits wide, then four cycles would be used to generate a comparand string that includes one or more (or none) of the bit groups from each of the 72 bit segments of the input string.

[0068] Selection circuitry 780 is coupled to receive the output of each of the program registers 790-791. One or more control signals may be applied on control line(s) 781 to selection circuitry 780 that selects among the outputs of the program registers 790-791. The control signals may be generated, for example,

by processor 310 of Figure 3A based on the operation cycles of the device, or by an instruction decoder or other control unit within the CAM device. The particular output of the program registers selected by selection circuitry 780 is applied to decode circuitry 740.

[0069] Decode circuitry 740 is coupled to receive data that is output from a respective program register to program switch 730 in a manner similar to that discussed above in relation to switch 430 of Figure 4A. Switch 730 operates to output one or more bits or group of bits of input string segment 705, received on input bus 735, into particular positions of comparand register 760. The operation of switch 730 is based on the data in the program registers decoded by decode circuitry 740. In order to not write over comparand data already stored in a particular comparand segment of comparand register 760 with translation information received in a later cycle, only the register positions that are to contain new data are written in any cycle through the use of the write enable control signals.

[0070] Note that the embodiments of the translation circuitry of Figures 4B, 5A and 5B may be used to implement the switch 730 of Figure 7.

[0071] In one embodiment, CAM device 320 may be partitioned into multiple blocks or block segments with each block and/or block segments capable of storing different tables for comparand lookups, as illustrated in Figure 6B.

10000158-103101

[0072] Figure 6B illustrates one embodiment of a CAM array having multiple blocks and multiple block segments. In one embodiment, CAM array 710 may be partitioned into two array blocks: block A and block B. The array blocks may be of the same or differing sizes. For example, block A may be 72 bits wide and block B may be 144 bits wide. Each block may have one or more block segments associated with it. For example, block A may include a segment 0 and block B may include block segments 1 and 2. Block 0 may store one lookup table and block segments 1 and 2 may store a different lookup table. Alternatively, different blocks may store the same lookup table.

[0073] As discussed above in relation to CAM 320 of Figure 3A, a block may be an entire array or a portion of a larger array. Although three block segments and two blocks are shown for ease of illustration, a CAM array may have more or less than three block segments and two blocks in alternative embodiments.

[0074] To operate with the multiple block configuration of Figure 6B, a CAM device may include a translation circuit (e.g., such as those described above) each corresponding to one of the blocks. One or more program registers in each of the translation circuitry may be programmed with translation information in order to generate the desired comparand strings for look-ups in blocks A and B, as discussed below in relation to Figure 7.

[0075] Figure 7 illustrates one embodiment of a CAM device having a multiple block CAM array and multiple translation circuitry. CAM device 700

may be CAM device 320 of Figure 3A. In one embodiment, CAM device 700 may include a CAM array 710 partitioned into multiple blocks that are organized into one or more arrays with each array and/or block capable of storing different tables for comparand lookups, as discussed above in relation to Figure 6B. As discussed above, a block may be an entire array itself, or a portion of a larger array and may include one or more smaller portions (e.g., block segments).

[0076] CAM device 700 also includes N number of translation circuitry (translation circuitry 0- translation circuitry N-1), each coupled to a corresponding CAM array block (block 0- block N-1). Each translation circuitry is shown as translation circuitry 715 of Figure 6B; however, any of the translation circuits described in this application may be used for one or more of the CAM array blocks.

[0077] Figure 8 is an illustration of the multiple cycle operation of a multiple array CAM device 700 of Figure 7. In the exemplary embodiment of Figure 8, CAM device 800 includes a CAM array 810 partitioned into array blocks 811 and 812. Array block 811 includes, for example, a single 72 bit block and Array block 821 includes two 72 bit block segments 822 and 823 (i.e., a 144 bit block). CAM device 800 also includes an input bus 835 having, for example, a 72 bit width. If an input string larger than 72 bits is to be applied to CAM device 800 (e.g., 288 bit input string 805), then all the bits of the input string cannot be applied simultaneously to CAM device 800 on input bus 835. Therefore, multiple operation cycles may be used to apply smaller segments of input string

805 on input bus 835, where each input string segment may have a maximum size of 72 bits. As such, for an input string having a total of 288 bits, four operation cycles may be used to apply input string segments to CAM device 800, where the input string segment for each cycle (cycle 1 segment, cycle 2 segment, cycle 3 segment, cycle 4 segment) is 72 bits. Cycle 1 segment includes 9 bytes (A0-A8), cycle 2 segment includes 9 bytes (B0-B8), cycle 3 segment includes 9 bytes (C0-C8), and cycle 4 segment includes 9 bytes (D0-D8).

[0078] PRs 851-854, 951-954, and 1051-1054 are utilized to generate the separate comparand strings stored in comparand registers 860, 960, and 1060, respectively, on different operation cycles of the device. On each operation cycle, PRs 851-854, 951-954, and 1051-1054 may be programmed to select any one of bytes  $A_0-A_8$ ,  $B_0-B_8$ ,  $C_0-C_8$ , and  $D_0-D_8$  for loading into any one of the byte positions of comparand registers 860, 960, and 1060. The byte data stored in comparand registers 860, 960, and 1060 may then be used to perform concurrent lookups in CAM block segments 811, 822, and 823, respectively. Alternatively, sequential look-ups may be performed in two or more of the block segments.

[0079] PRs 851-854, 951-954, and 1051-1054 may be similar to the program register described above. For example, PRs 851-854, 951-954, and 1051-1054 may each be a nine position register with each register position storing translation information that correlates to a particular byte of input string 805. The information may be stored, for example, in binary format. With such a format, only four bits are needed in each position to reference a particular byte of input

string 805 and/or generate a write enable signal. For example, a 0000 binary bit pattern may be used to designate byte 0. As such, each of PRs 851-854, 951-954, and 1051-1054 are 36 bit registers in this exemplary embodiment. In an alternative embodiment, the PRs in each translation circuitry (e.g., PR 851-854) may not be separate registers but, rather, sections of one or more larger registers.

[0080] If the comparand strings  $A_0A_1B_0B_1B_8D_1$ , and  $A_5A_6A_8B_1B_2B_3D_0D_1D_2D_3D_4D_5A_0A_1$  are desired for performing lookups in array blocks 811 and 812 (i.e., block segments 822 and 823) respectively, then such comparand strings may be loaded into comparand registers 860, 960, and 1060 respectively, on four cycles of operation by programming PRs 851-854, 951-954, and 1051-1054 accordingly. Alternatively, the comparand string segment for lookup in array block 812 may be considered as two distinct string segments  $A_5A_6A_8B_1B_2B_3D_0D_1D_2$ , and  $D_3D_4D_5A_0A_1$  corresponding to lookups block segments 822 and 823, respectively.

[0081] For ease of discussion, the loading of comparand registers 860, 960, and 1060 will each be discussed separately, below. It should be noted, however, that the components associated with the loading of comparand registers 860, 960, and 1060 may operate concurrently on each given cycle. For example, decode circuitry 840, 940, and 1040 concurrently decodes the translation information programmed in PRs 851, 951, and 1051, respectively, during cycle 1; decode circuitry 840, 940, and 1040 concurrently decodes the translation information programmed in PRs 852, 952, and 1052, respectively, during cycle 2; decode



circuitry 840, 940, and 1040 concurrently decodes the translation information programmed in PRs 853, 953, and 1053, respectively, during cycle 3; and decode circuitry 840, 940, and 1040 concurrently decodes the translation information programmed in PRs 854, 954, and 1054, respectively, during cycle 4. In addition, bytes of input string segments 801-804 may be concurrently loaded into comparand registers 860, 960, and 1060, respectively, on a particular cycle, after the translation operation.

[0082] With regard to the loading of comparand register 860 with bytes  $A_0A_1 B_0 B_1 B_8 D_1$ : PR 851 may be programmed with the following pattern in its nine positions -- 0 1 X X X X X X X; PR 852 may be programmed with the pattern -- F F 0 1 8 X X X X; PR 853 may be programmed with the pattern -- F F F F F X X X; and PR 854 may be programmed with the pattern F F F F F 1 X X X.

[0083] As previously discussed, the symbols in each register position represents a particular byte number of an input string. As such, the first byte position of PR 851 may actually be storing the bits 0000 that designates byte 0 in a binary format. The symbol X represents a don't care condition where the particular byte position may be overwritten in subsequent cycles and eventually globally masked (or locally masked if the CAM cells are ternary CAM cells) before transmission to the CAM array blocks as discussed above. Alternatively, all the X's may be replaced with F's. F represents a code that instructs decode circuitry to inhibit a write operation. F, in this particular embodiment, represents all 1s, but any other code may be used. Therefore, on the first cycle, a

control signal applied to the control input 881 of multiplexer 880 configures multiplexer 880 to select particular bits (e.g., bytes) of the input string 805 designated by the contents of cycle 1 PR 851 for output to decode circuitry 840.

[0084] Decode circuitry 840 and switch 830 may operate in a manner similar to that discussed above for decode circuitry 440 and switch 430, to decode the contents of PR 851 and load particular bytes of the input string segment 801 into particular register positions of comparand register 860. With the first position of PR 851 containing a byte 0 designation, decode circuitry 840 instructs switch 830 to load byte 0 of segment 801 (i.e., byte A0) into the first position 861 of comparand register 860. With the second position of PR 851 containing a byte 1 designation, decode circuitry 840 instructs switch 830 to load byte 1 of segment 801 (i.e., byte A1) into the second position 862 of comparand register 860.

[0085] On the second cycle, multiplexer 880 may be configured to select particular bits (e.g., bytes) of the input string 805 designated by the contents of cycle 2 PR 852 for output to decode circuitry 840. With the first two positions of PR 852 containing an F bit code, switch 830 is inhibited from writing to bit positions 861 and 862 of comparand register 860. As previously discussed, decode circuitry, for example, decode circuitry 840 may be configured to control when the comparand register 860 is loaded. The write enable signal may be generated based on the decoding of an F code by decode circuitry 840 and operates to control which positions of comparand registers 860 are written to.

10000158 "103101

[0086] In this manner, byte A0 and A1 stored in bit positions 861 and 862, respectively, of comparand register 860 are not over-written. The inhibiting of write operations is known in the art; accordingly a detailed discussion is not provided. With the third position of PR 852 containing a byte 0 designation, and the fourth position of PR 852 containing a byte 1 designation, and the fifth position of PR 852 containing a byte 8 designation, decode circuitry 840 instructs switch 830 to load: byte 0 of segment 802 (i.e., byte B0) into the third position 863 of comparand register 860, byte 1 of segment 802 (i.e., byte B1) into the fourth position 864 of comparand register 860, and byte 8 of segment 802 (i.e., byte B8) into the fifth position 865 of comparand register 860. The X designation in the remaining positions of PR 852 may be globally masked after the loading of comparand register 860 prior to transmitting the contents of comparand register 860 to the array 811

[0087] On the third cycle, multiplexer 880 may be configured to select particular bits (e.g., bytes) of the input string 805 designated by the contents of cycle 3 PR 853 for output to decode circuitry 840. Because the desired comparand string to be stored in comparand register 860 does not contain any bytes from input string segment 803, the bit positions of PR 853 contain F designations to prevent the over-writing of previously written to positions of comparand register 860 and X designations of all other bit positions.

[0088] On the fourth cycle, multiplexer 880 may be configured to select particular bits (e.g., bytes) of the input string 805 designated by the contents of

cycle 4 PR 854 for output to decode circuitry 840. With the first five positions of PR 854 containing an F designation, switch 830 is inhibited from writing to bit positions 861-865 of comparand register 860. In this manner, bytes  $A_0$ ,  $A_1$ ,  $B_0$ ,  $B_1$ ,  $B_8$  stored in bit positions 861-865, respectively, of comparand register 860 are not over-written. With the sixth position of PR 854 containing a byte 1 designation, decode circuitry 840 instructs switch 830 to load byte 1 of segment 804 (i.e., byte D1) into the sixth position 866 of comparand register 860. The remaining positions 867-869 (e.g., containing don't cares) do not participate in the subsequent lookup and may be masked out by one or more global mask registers.

[0089] In a manner similar to that discussed above, comparand register 960 may be loaded with bytes  $A_5$ ,  $A_6$ ,  $A_8$ ,  $B_1$ ,  $B_2$ ,  $B_3$ ,  $D_0$ ,  $D_1$ ,  $D_2$  of input string segments 801-804 by programming: PR 951 with the pattern -- 5 6 8 X X X X X X; PR 952 with the pattern -- F F F 1 2 3 X X X; PR 953 with the pattern -- F F F F F F X X X; PR 954 with the pattern -- F F F F F F 0 1 2.

[0090] Continuing the example, comparand register 1060 may be loaded with bytes  $D_3$ ,  $D_4$ ,  $D_5$ ,  $A_0$ ,  $A_1$  of input string segments 801-804 by programming: PR 951 with the pattern -- X X X 0 1 X X X X; PR 952 with the pattern -- X X X F F X X X X; PR 953 with the pattern -- X X X F F X X X X; PR 954 with the pattern -- 3 4 5 F F X X X. As illustrated by the exemplary pattern loaded in comparand register 1060, bytes from input string segment received in a later cycle (e.g., byte  $D_3$  from

segment 804) may be loaded into either upper most or lower most positions of the comparand register (e.g., byte D<sub>3</sub> loaded into position 1061).

[0091] It should be noted that the resulting comparands loaded into comparand registers 860, 960, and 1060 may have different widths. For example, the comparand loaded into comparand register 860 is six bytes wide, the comparand loaded into comparand register 960 is nine bytes wide, and the comparand loaded into comparand register 1060 is 5 bytes wide. Alternatively, comparands of the same widths may be generated and loaded into various comparand registers.

[0092] As previously discussed, decode circuitry 840, 940, and 1040 may also be configured to output a write enable (WE) signal to control the loading of comparand registers 860, 960, and 1060, respectively. The write enable signal operates to control when positions of comparand registers 860, 960, and 1060 are written to based on the decoding of the translation information by decode circuitry 840, 940, and 1040, respectively (e.g., write when the translation information is not an F code).

[0093] It should also be noted that, the comparands generated by the translation circuitry described in the various embodiments above, may have gaps (e.g., in their register positions such that they are loaded with non-contiguous data) and/or have repeated bits or bit groups. In one embodiment, particular input string bits or bit groups (e.g., byte A<sub>0</sub> of Figure 8) may be translated to be in multiple positions of the same comparand.

1000158-103101

[0094] Note that the embodiments of Figures 6A, 7 and 8 may also use to the translation circuitry of Figure 5B.

[0095] By decoding a common input string to generate multiple comparands in parallel, rather than sequentially, concurrent lookups on separate CAM arrays may be performed. Thereby, packet throughput in a CAM may be significantly increased. It should be noted that the number and sizes of the components and cycles of Figure 8 are only exemplary and that other configurations for a CAM device 700 may be used.

[0096] In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense.

10000158-103101